Review of Weight Uncertainty in Neural Network

Some useful links:

1. Paper can be found here
2. Implementation by Pytorch on Github

Bayes by Backprop (BBP) Framework

A probabilistic model: $P(y|x,w)$: given an input $x \in \mathbb{R}^p, y \in \mathcal{Y}$, using the set of parameters or weights w.

$$P(w|x,y) = \frac{P(y|x,w)p(w)}{P(y)}$$

Loss Function

The weights can be learnt by MLE given a set of training samples $\mathcal{D} = \{x_i, y_i\}_i$

$$\begin{aligned} w^{MLE} &= \arg\max_w \log P(\mathcal{D}|w) \\ &= \arg\max_w \log \sum_i^n P(y_i|x_i,w) \end{aligned}$$

Regularization can be done by add a prior on the weights w and finding the MAP, i.e.,

$$\begin{aligned} w^{MAP} &= \arg\max_w \log P(w|\mathcal{D}) \\ &= \arg\max_w \log P(\mathcal{D}|w)p(w) \end{aligned}$$

Inference is intractable because it needs to consider each configuration of w.

$$P(\hat{y}|\hat{x}) = \mathbb{E}_{p(w|D)}[P(\hat{y}|\hat{x},w)]$$

Minimization of KL Divergence (ELBO)

$$\begin{aligned}\theta^* &= \arg\min_\theta KL[q(w|\theta)|P(w|\mathcal{D})] \\ &= \arg\min_\theta \int q(w|\theta)\log\frac{q(w|\theta)}{P(w|\mathcal{D})}dw \\ &= \arg\min_\theta \int q(w|\theta)\log\frac{q(w|\theta)}{P(w)P(\mathcal{D}|w)}dw \\ &= \arg\min_\theta \underbrace{KL[q(w|\theta)|P(w)]}_{\text{complexity cost}} - \underbrace{\mathbb{E}_{q(w|\mathcal{D})}[\log P(\mathcal{D}|w)]}_{\text{likelihood cost}}\end{aligned}$$

We denote it as:

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) &= KL[q(w|\theta)|P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}|w)] \qquad (1) \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\log q(w^i|\theta) - \log P(w^i) - \log P(\mathcal{D}|w^i) \qquad (2)\end{aligned}$$

where w^i is a sample from variational posterior $q(w|\theta)$. Note that the parameters require gradient is θ instead of w in MLE or MAP. Note that in the original paper there is no $\frac{1}{n}$. I think it is probably a typo.

Unbiased Monte Carlo Gradients

> Proposition 1. Let ϵ be a random variable having probability density given by $q(\epsilon)$ and let $w = t(\theta,\epsilon)$ where t is a deterministic function. Suppose further that the marginal probability density of w, $q(w|\theta)$, is such that $q(\epsilon)d\epsilon = q(w|\theta)dw$. Then fora a function f with derivative in w: $\frac{\partial}{\partial\theta}\mathbb{E}_{q(w|\theta)}[f(w,\theta] = \mathbb{E}_{q(\epsilon)}[\frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial\theta} + \frac{\partial f(w,\theta)}{\partial\theta}]$

Our objective function in Eq. (1) can be written as

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) &= \mathbb{E}_{q(w|\theta)}[\log q(w|\theta) - \log P(w) - \log P(\mathcal{D}|w)] \\ &= \mathbb{E}_{q(w|\theta)}[f(w,\theta)]\end{aligned}$$

We need the gradient

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{q(w|\theta)}[f(w,\theta)] &= \nabla_\theta \int f(w,\theta) q(w|\theta) dw \\
&= \nabla_\theta \int f(w,\theta) q(\epsilon) d\epsilon \\
&= \mathbb{E}_{q(\epsilon)}[\nabla_\theta f(w,\theta)]
\end{aligned}
$$

Leibniz integral rule can also used here.

Why re-parameterization trick

Some useful links:

1. Introduction about MC gradient
2. Why re-parameterization trick

Normally, we use Monte Carlo (MC) gradient estimator to approximate the gradient. It is used to solve the problem $\nabla_\theta \mathbb{E}_{q(w)}[f(w,\theta)] = \mathbb{E}_{q(w)}[\nabla_\theta f(w,\theta)]$. **Note that there is no parameter θ in the expectation distribution $q(w)$, which is the difference between this case and the case we are facing.**

If we still use MC gradient estimator on $\nabla_\theta \mathbb{E}_{q(w|\theta)}[f(w,\theta)]$:

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{q(w|\theta)}[f(w,\theta)] &= \int \nabla_\theta [q(w|\theta) f(w,\theta)] dw \\
&= \int \nabla_\theta [q(w|\theta)] f(w,\theta) dw + \int q(w|\theta) \nabla_\theta [f(w,\theta)] dw \\
&= \int q(w|\theta) \nabla_\theta [\log q(w|\theta)] f(w,\theta) dw + \mathbb{E}_{q(w|\theta)}[\nabla_\theta [f(w,\theta)]] \\
&= \mathbb{E}_{q(w|\theta)}\left[\nabla_\theta [\log q(w|\theta)] f(w,\theta)\right] + \mathbb{E}_{q(w|\theta)}[\nabla_\theta [f(w,\theta)]].
\end{aligned}
$$

The problem is that the distribution in the first term is coupled with both w and θ. The value of the first term also depends on θ but θ is what we are tunning. We need to detach θ from the distribution. Then, we can apply the MC gradient estimator.

Why $q(\epsilon) d\epsilon = q(w|\theta) dw$?

For deterministic mapping $w = t(\epsilon, \theta)$, $q(\epsilon) d\epsilon = q(w|\theta) dw$ holds.

$$
\begin{aligned}
q(w|\theta)\frac{dw}{d\epsilon} &= q(\epsilon) \\
q(w|\theta) &= Kq(\epsilon) \\
G(\epsilon) &= Kq(\epsilon)
\end{aligned}
$$

Leibniz integral rule

> Leibniz integral rule:
> $\frac{d}{dx}\left(\int_{a(x)}^{b(x)} f(x,t)\,dt\right) = f\big(x,b(x)\big)\cdot\frac{d}{dx}b(x) - f\big(x,a(x)\big)\cdot\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{\partial}{\partial x}f(x,t)\,dt$

Mini-batches

For each epoch of optimization, the training set is equally and randomly split into M batches $\mathcal{D}_1, ..., \mathcal{D}_M$. The loss can be rewritten as

$$\mathcal{F}_i(\mathcal{D},\theta)_= \frac{1}{M}KL[q(w|\theta)|P(w)] - \mathbb{E}_{q(w|\theta)}[\log P(\mathcal{D}|w)]$$

I'll update more details later.